Machine Learning for Physicists

Predicting Confirmed Cornona Cases Based on a Dataset of Population, Temperature and Air Pressure

Philipp Zolthoff
philipp.zolthoff@tu-dortmund.de

July 25, 2023

TU Dortmund – Fakultät Physik

Contents

1 Motivation

The last years were heavily defined by the defestating results of the COVID 19 pandmic [1] and thus we were greatly impactaed by that viurs. One of the most dominant factors of every new desease is its growth i.e the confirmed cases in a given population. A most common approach to this is relying on the exponential groeth, and thus fitting it this way. The main idea of this project is to find an efficient way to predict the confirmed cases of COVID 19 patients through a selection of features that might corrolate. Since COVID is transmitted though every day contact via aerosols [1] the features will be a selection of weather data containing temperatures and air pressure. It is the idea to look for dependencies of, for example, good weather that might lead to an increase outgoing of people and thus a higher spread of the virus or the opposite case, where bad weather would weaken the immunn system and thus increase the general test rate with a resulting increas of confirmed cases. A geological feature will not be included to isolate the problme from social status and other political factors. To fully understand the impact of the weather it is of great importance to also include its time dependencies since four days of bad waether might have a different impact than an alternating weather cycle. Thus this project will aim to train a recurrent neural network [4] to satisfy the time sequenced data with the goal to predict the confimred corona cases of several cities.

2 Used Datasets

The datasets used for this project, inlcuding three main sets of data, make up the informations for a total of 8 features. Since it will be the goal to predict COVID 19 cases, the data must contain confirmed cases on the geological smallets possible scale, for that if only countries are compared the data will be reduced to approximatly 150 entries. For this project the data will come from the johns hopkins university which provides a well tracked essamble of confrined corona cases on a city scale.

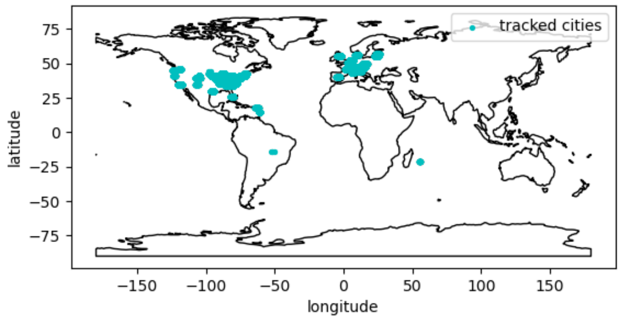


Figure 1: Shown is a schematic overview of the cities used to predict confimred corona cases using geopandas [2].

Thus an unproccesed amount of approximatly 20000 cities, coded through latitude and longitude, with the features of interest are obtained. To collect worldwide weather data, again coded through latitude and longitude, the opensource platform "Metostat" [3] will bee used. They provide a vast amount of features from which the project will use the air pressure, avarage -, maximal -and minimal-temperature of each day. In a first step of merging both sets will be transformed to a “GeoPandasDataFrame”[2] which creates a new “Geometry” feature, containg botch longitude and latitude. With a choosen margin, in geopandas this is called “buffer”, the entries will be joined with its representing counterpart that is close i.e. in the margin of the first called buffer. For this a buffer of 0.005 has been choosen to allow some deviation since the geological datapoints of the combined sets might not be taken at the exact same location. The final Datafram will

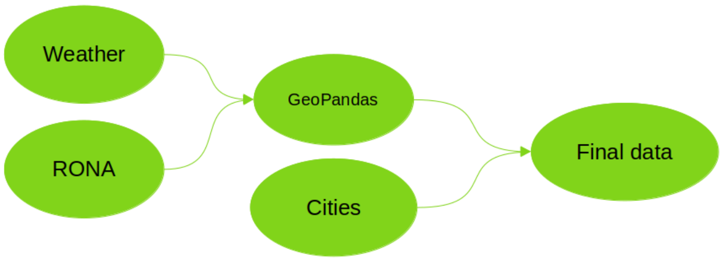


Figure 2: Shown is a schematic overview on how three different Dataset are used to form one final set with the desired features.

inlcude the features: confirmed cases, minimal Temperature, maximal Temperature, average Temperature, air pressure, date and population. Note that the date will be transformed into seconds, then divided by 10^{-8} rather than days months and years.

confirmed	tmin	tmax	tavg	pres	date	city	population
0	-5.5	5.0	-1.2	1030.9	1.579651	1294	10973.0

Figure 3: Exampletory slice of the dataframe used to predict confirmed CONVID cases.

The scatter between the different features, including confirmed cases, can be seen in **??**. An obvious linear correlation between “T_min”, “T_max” and “T_avg” can be seen yet also an almost exponential growth in the confirmed-date cell.

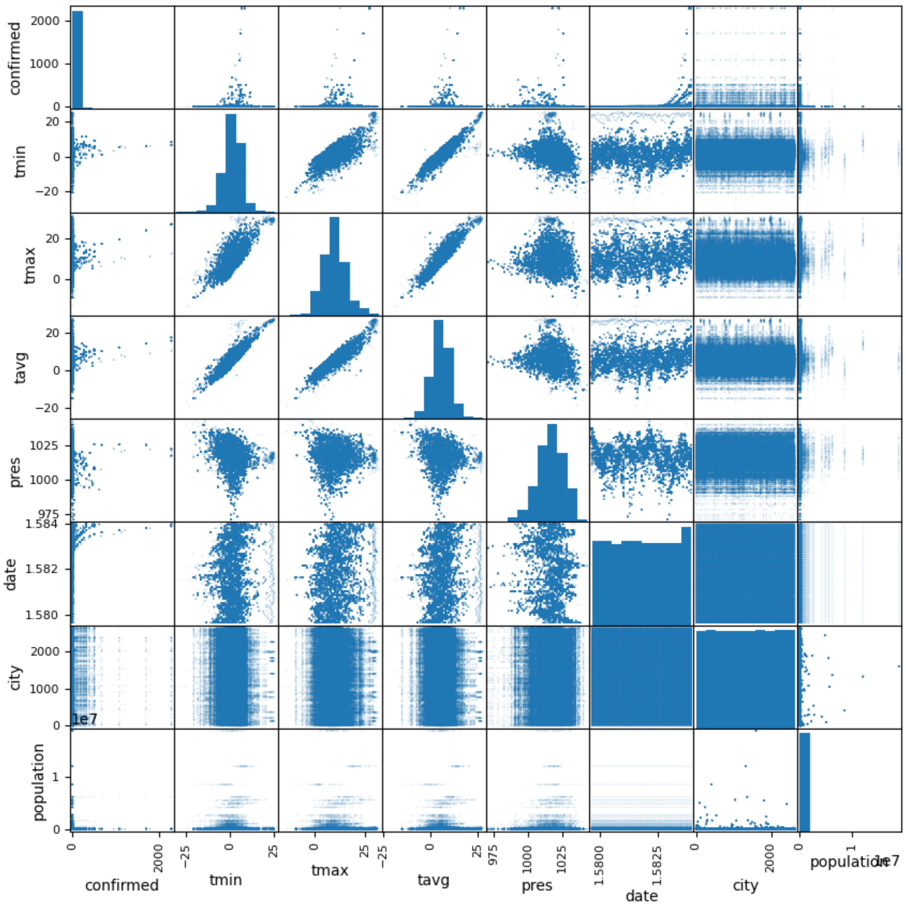


Figure 4: This graphic shows a scatter matrix of a dataset that carries several features related to both weather and COVID confirmed cases.

3 The Solution

3.1 Data Preperation

The problem that the neural network is trained for relies on time depended data, i.e. time based weather data where the order of events does have an impact on the outcome. To include that, recurrent neural networks, short RNN, will be build the ground for a first approach. As an eviroment to build the model in, “Keras” has been choosen as it provides all the tools, layer structures and evaluation methods to tackle the problem at hand [4]. At first the dataset has to be further prepared and sorted that the outcome shape will divide every city, with each city including 51 days of 8 features. It follows that one entry contains 51 time sequenced rows with informations. A test train split, provided by “Keras” [4], will randomly seperate the data into different regimes: 20% test, 64% train, 16% validation. Each regime will be split in two, where the first 40 entires, i.e. days, of each city will be the input of the model and the last 11 entries of confirmed cases (shape (11,1)) the output, i.e. target. To handle outlier and make the sets more handable for the umpcoming network a “Standardscaler” [4] will be applied to each of them.

3.2 The Model

Since the already foreseen use of RNNs, three simple models have been constructed each with one layer containing: “SimpleRNN”, “GRU” and “LSTM” to get a quick overview on how each RNN architecture perfoms on this problem. The “simpleRNN”

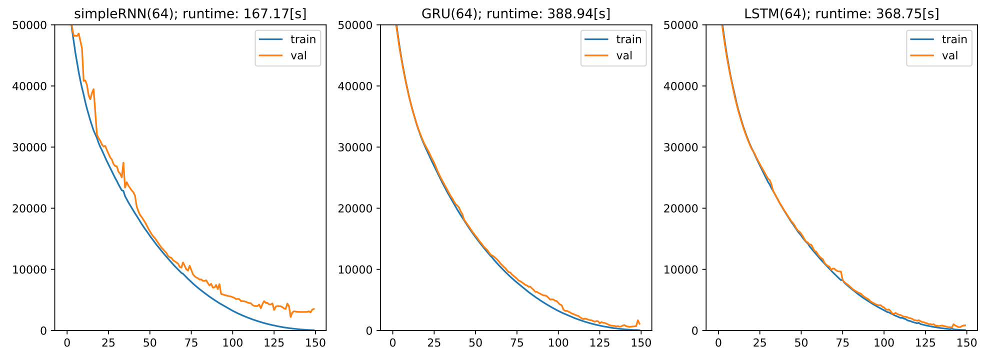


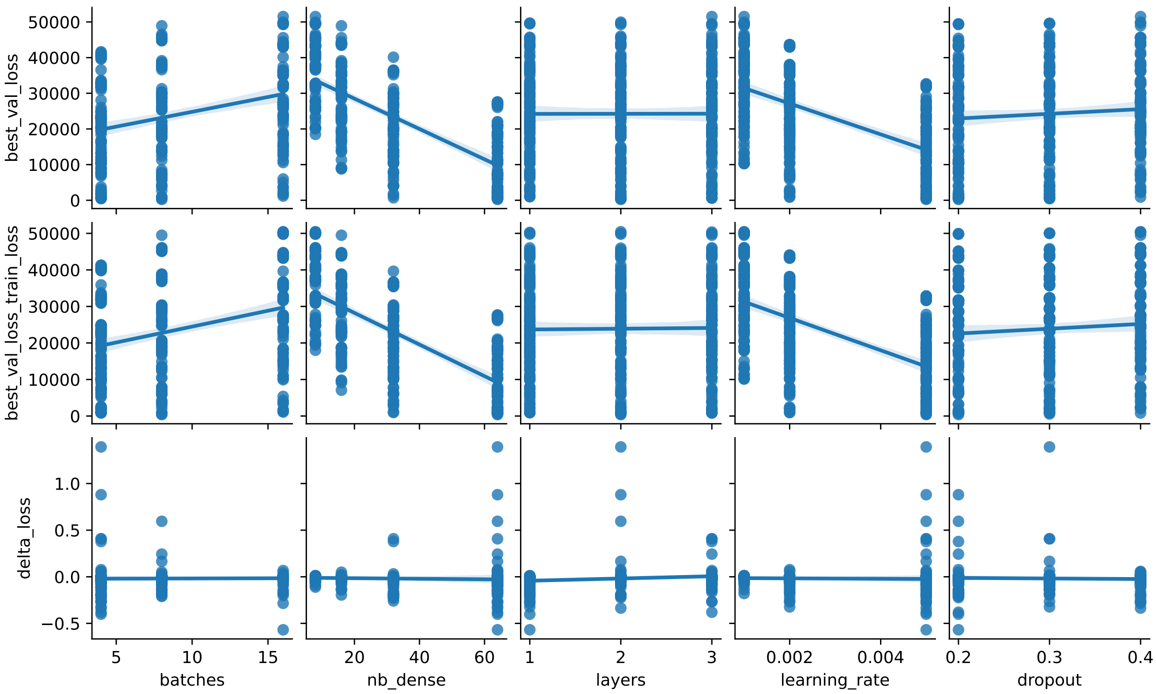
Figure 5: Displayed are three diferent RNN layers, as loss functions, trained on the same dataset with the representing runtime in seconds.

loss function in 5 shows a dispersion between the train and validation indicating overtraining, yet a similar result to the more sufficicated layers. Both “GRU” and “LSTM” take approximatly the same time for 150 epochs with similar results. In the following “SimpleRNN” will be choosen, with “adam” as an optimizer [4], as the main layer architecture for its short runtime. This first selection has been done with only one layer as a simple representation.

3.3 Finetuning SimpleRNN

To apply a fundamental structore to the model a gridsearch, span over an amount of 324 combinations, will look for the best aoutcome over an epoch count of 150, inlcuding an “early stop” mechanism with a patience of 10. The gridsearch will search for fititng candidates for: learning rate, units in each layer, layer count, dropout magnitude and batch size. For this each layer represents one layer of SimpleRNN architecture.

3.4 Gridsearch Resuluts



	layers	nb_dense	dropout	learning_rate	batches	best_val_mae_train_mae	best_val_mae	best_val_loss_train_loss	best_val_loss	best_val_epoch
301	2	64	0.2	0.005	8	6.307665	4.993853	390.146729	244.596573	135
307	2	64	0.3	0.005	4	8.689884	6.171397	1119.690186	467.970978	92
310	2	64	0.3	0.005	8	6.820852	6.264912	556.710938	477.608124	126
298	2	64	0.2	0.005	4	10.660253	6.181417	975.801147	518.952271	97
308	3	64	0.3	0.005	4	8.314013	6.522310	838.901123	596.157043	102

Figure 6: Displayed is a hyperparameter optimization via gridsearch to find optimal parameters for: learning rate, dropout, numbers of layers and batch size. The top graphic displays a "pairplot" [5] inlcuding a linear fit to predict the models correlation towards a ceratin hyperparameter. The bottom grahpic shows the first five entries of the presented gridsearch, presententing the found parameters, sorted by "best_val_loss".

The model picked for further finetuning is the third one in the bottom of 6 since it provides a good validation loss without a big discreptancy towords the training loss which only indicates low overtraining when compared to different models (found by the gridsearch). After the application of gridsearch some adjustments will be tried out manually with a focus on the learning rate. For that the optimization learned with a constant rate, a "learning rate schedualer" will be implemened as a further callbackfunction that automaticly activates himself after a given amount of epochs. The learning rate then decreases in an exponential behaviour, which argument can be further finetuned to acomplish a smooth convergence against the best possible loss rate. For the tuned model an epoch of 100 will be choosen, with an argument of

-0.05 in the exponential funciton, since from that point the validation loss struggles to converge. As the last layer is a dense model with 11 outputs, i.e. the 11 days the model has to predict, a variation of saclers and outputfunctions can be applied. Various scaler do not perform as good as the unsacled target in combination with a "softplus" activation function, where

$$\text{softplus}(x) = log(exp(x) + 1). \tag{1}$$

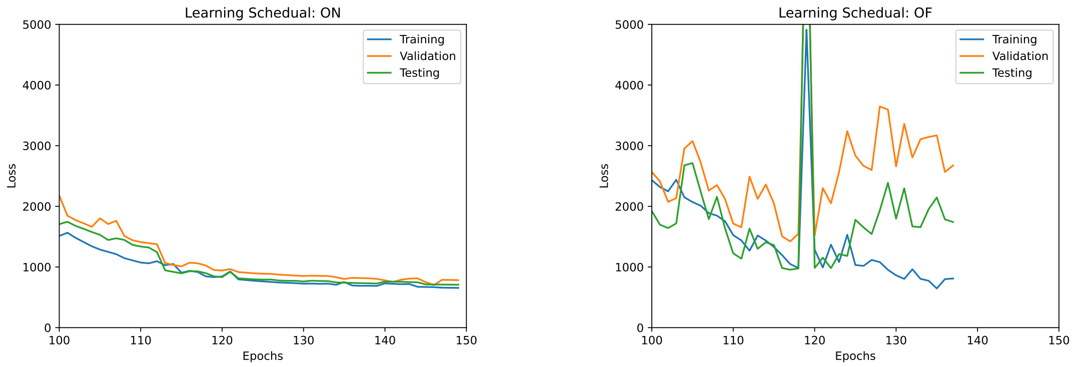


Figure 7: Displayed are two loss curves, both for the same dataset, trained on the same model, but with an activated leanring schedualer on the left side. The right side displays the same model without the modified learning rate.

In 7 can bee seen how the loss function converges with a dynmic learning rate rather than a constant learning rate where the supposibly local minimum is not found. The same results in the left graphic can also be archieved through lowering the constant learning rate to a certain amount, so the minium will be approached more caregfully, yet this would take more epochs and thus more runtime to acomplish.

3.5 Keeping Overfitting In Check

To avoid overfitting, several mechanism and imported tools will be applied to the model. For that it is an indicator of overfitting when the valdiation loss converges yet the taring loss still decreases, the callbackfunction "early stop" [4] will be applied with a patience of 10 that monitors the validation loss and stops if the current validation loss will not be undercut after 10 epochs. Furthermore for each layer of LSTMs one layer of dropout will force the units to be trained in a general way so that no unit specillizes for the training set.

4 Alternativ approach

As an alternative approach the routine of finding the least square for a given function will be choosen. With the approximatly exponential behviour of most pandemics, when it comes to confirmed cases, the fit will be applied to the form: $a \cdot exp(b \cdot t) + c$ where t is the time in days and a, b, c trainable parameters. Since the alternative method will not rely on a train data set, but on the 40 days ahead of the 11 that are about to be predicted, only the test data will be of use here. For that every cities 40 first days will be fitted with an exponential function against the confirmed cases, followed by calculation of the mean squared error, of the then predicited following 11 days, to be later compared against the nerual net. In this context the alternative approach will rely on the size of the onterval of last days to be predicted. Increasing or decreasing the last 11 days as a target will thus make the resulting error smaller, not only because of less points to predict but the more data that trains the fit.

5 Results

The results contain both the model and a comparison of the model against the alternative approach for that it is the gaol of the model to outperfom the classic technice, not inlcoding neural networks.

5.1 Evaluation Of The Model

The trained model, when applied on the test data, shows a final loss of 859.59 and a mae of 8.43. Since the training process was monitored by the mse rather than the mae, the mse will be the leading factor in this evalulation. Mse has been choosen for its abbilty to inlcude outliners more into its error in comparison with mae. This characteristic, in this context, is more desired to surpress abnormalities, i.e. focus on the general increase of confirmed COVID cases.

5.2 The Model vs. The Alternativ Approach

The model trained for this report and the alternative approach vary in both training concept and results. While the nerual net profits from an test/train split to train on, the least-square-fit-routine is directly applied to the first 40 days of the test dataset and thus does not experience the rest of the data. The mse of the alternative approach reads 19944936.25. The figure in 5 shows the growing deviation of test data towords the end, i.e. the last days, where the exponential function strongly increases with every day. For the neural net already knows on how the last 11 days can look like, because it was trained on that with the test and validation set, its error does not increase at a later period.

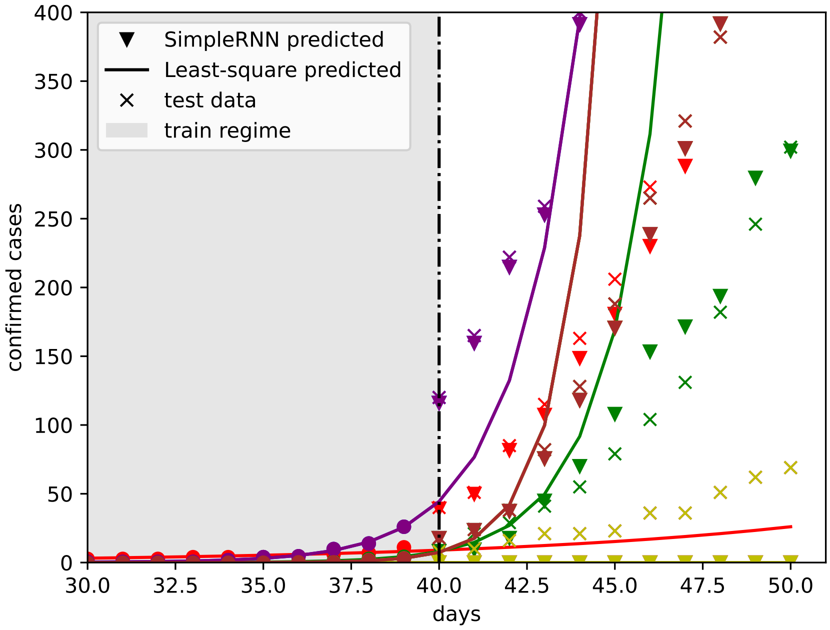


Figure 8: Displayed is a comparison of predictions on confirmed COVID cases between a least square fit routine and a trained neural net. The days are cut to the last 20 days, only including cities where confirmed cases have been found. Two different declarations of confimred cases are given: SimpleRNN predictions and exponential fitted predictions, while the test data is marked with an x.

6 Conclusion

For the task, predicting confirmed COVID cases after a given 40 days of data inclding weather informations, the nerual net outperfoms the alternative approach by several magnitudes ($mse_{alternative} = 19944936.25 \quad \rightarrow \quad mse_{SimpleRNN} = 859.59$).
The alternative method haevily relies on the amount of days it is fitted with while the SimpleRNN seemingly knows how to predict a certain coure based on the weather and confirmed cases before the prediction. Even tho the neural net perfoms better, it still gets some predictions wrong as can be seen for the yellow case in 8. An other approach to the model would be to train the data on a more suffisticated RNN architecture for example “LSTM”, which would take more time and recources, yet might perfom better on the problem at hand.

References

[1] Ensheng Dong, Hongru Du, and Lauren Gardner. "An interactive web-based dashboard to track COVID-19 in real time." In: *The Lancet Infectious Diseases* 20.5 (2020), pp. 533–534. ISSN: 1473-3099. DOI: https://doi.org/10.1016/S1473-3099(20)30120-1. URL: https://www.sciencedirect.com/science/article/pii/S1473309920301201.

[2] Kelsey Jordahl et al. *geopandas/geopandas: v0.8.1*. Version v0.8.1. July 2020. DOI: 10.5281/zenodo.3946761. URL: https://doi.org/10.5281/zenodo.3946761.

[3] Christian Sebastian Lamprecht. *Meteostat Python*. URL: 'https://orcid.org/0000-0003-3301-2852'.

[4] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[5] Michael L. Waskom. "seaborn: statistical data visualization." In: *Journal of Open Source Software* 6.60 (2021), p. 3021. DOI: 10.21105/joss.03021. URL: https://doi.org/10.21105/joss.03021.

Appendices

Code With Attached Link To Google Colab